Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electronic Notes in Theoretical Computer Science 149 (2006) 61–77
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

Algorithmic Algebraic Model Checking III: Approximate Methods

Venkatesh Mysore[1,2]

Computer Science Department, Courant Institute
New York University, New York, USA

Bud Mishra[3]

Departments of Computer Science and Mathematics, Courant Institute
Department of Cell Biology, School of Medicine
New York University, New York, USA

Abstract

We present computationally efficient techniques for approximate model-checking using bisimulation-partitioning, polyhedra, grids and time discretization for *semi-algebraic hybrid systems*, and demonstrate how they relate to and extend other existing techniques.

Keywords: Semi-algebraic hybrid systems, Model checking.

1 Introduction

A *semi-algebraic hybrid automaton* [9,8] is a hybrid automaton, whose expressions corresponding to the initial values, state invariants, continuous flows, and the guards and resets of the discrete transitions are all semi-algebraic, i.e., Boolean combinations of polynomial equations and inequalities. They are often used to approximate more general systems, whose flow equations are not

[1] The work reported in this paper was supported by grants from NSF's ITR program and DARPA's BioCOMP program.
[2] Email: mysore@cs.nyu.edu
[3] Email: mishra@nyu.edu

doi:10.1016/j.entcs.2005.11.017

polynomial, since truncated Taylor series, polynomial splines and other symbolic integration schemes provide good semi-algebraic local approximations for flows, etc. A *location* of a semi-algebraic hybrid automaton H is a pair $\langle v, X\rangle$, where $v \in V$ is a state and $X \in \mathbb{R}^k$ is an assignment of values to the k system variables. The *transition relation* $\langle u, X\rangle \xrightarrow[T]{h} \langle v, X'\rangle$ of H connects all possible values of the system variables before and after *one step*; namely, it is either a discrete step $\langle u, X\rangle \xrightarrow[\mathcal{D}]{0} \langle v, X'\rangle$ for a time $h = 0$ or a continuous evolution $\langle u, X\rangle \xrightarrow[\mathcal{C}]{h} \langle v(= u), X'\rangle$ for a time period $h > 0$. (For detailed definitions and summary of results, please refer to the *Appendix* .)

Earlier, in the first paper in this series [9], we introduced this class and demonstrated the use of real algebraic methods for solving the bounded reachability problem. In the second paper [8], we examined the *single-step until* operator $p \triangleright q$ of the dense-time logic *Timed Computation Tree Logic* (TCTL), which is defined as $p \vee q$ holding all along "one step" of the hybrid system and q being true at the end of the transition. *Since quantifier elimination over semi-algebraic sets is decidable [10], $p \triangleright q$ was shown [8] to be decidable for semi-algebraic hybrid systems if p and q were also semi-algebraic.* It was further proved [8] that the "existential" segment of TCTL (including reachability) and the negation of the "universal" segment are semi-decidable over semi-algebraic hybrid automata. Further, all subscripted TCTL operators become decidable in the absence of zeno-paths.

Effectively, the symbolic algebraic model checking problem was reduced to a series of quantifier elimination problems which could be solved by a software tool such as Qepcad [5]. The only source of error (if any) arose in approximating non-polynomial systems. However, the computational complexity (double exponential) of the cylindrical algebraic decomposition severely limited the applicability of the method. In this paper, we discuss the applicability of the different approximation approaches involving space- and time- discretization to semi-algebraic hybrid automata. Approximate methods have been very successful in timed automata and linear hybrid systems, yielding efficient decidable algorithms in many cases [3,4,2,1]. However, these methods rely on computational techniques that exploit the low dimensionality and other restrictions, of the dynamics of these subclasses of hybrid systems. In other words, the techniques are seldom applicable to more complex systems. Our first goal in this paper is to show that many existing ideas can be made applicable to semi-algebraic hybrid systems, by using quantifier elimination in place of the original efficient-but-restrictive computational method. The second goal is to develop these ideas to obtain new optimizations and techniques. Further, we seek to identify well-behaved sublcasses that are more general

than timed or linear automata.

By suitably relaxing accuracy requirements, we aim to model-check the vast semi-algebraic class, without being severely computationally hindered. Quantifier elimination will still remain our engine of computation, though it will be used differently; namely, it will be invoked to solve many simple problems instead of a few complex problems. In this paper, we will present the modified versions of existing techniques and understand their behavior over the semi-algebraic class. Clearly, different techniques will prove to be effective in different scenarios. In this paper, we do not delve into this aspect, but instead focus on generalizing and optimizing existing techniques. In this sense, it is the first effort to catalogue the algorithms for approximate verification (reachability) for the vast semi-algebraic class.

2 Approximate Methods

In this section, we develop new approximation methods applicable to the semi-algebraic class, based on the existing literature for much simpler subclasses of hybrid automata. For brevity, all proofs are provided in the *Appendix*.

2.1 Bisimulation Partitioning

The bisimulation idea is to convert the given hybrid automaton into a simpler one, which only preserves the properties of interest to us (in the query). The conventional bisimulation partitioning algorithm [4] involves splitting the discrete states based on the out-going discrete transitions. The source state of a transition is split so that, each new state (its partitions) has the minimal number of out-going transitions (ideally, each new partition will have only one possible successor discrete state). The rationale is that one expects only some of the guards (of the different out-going transitions) to be satisfiable, from different parts of the continuous space representing the discrete state (its state invariant).

We first prove that these partitions are *computable* for semi-algebraic hybrid systems by expressing the task as a quantifier elimination problem.

Theorem 2.1 *The standard bisimulation partitions are computable for semi-algebraic hybrid automata.* □

Having proved that the existing idea becomes applicable via quantifier elimination, we now suggest an improvement of the technique. This approach is founded on the observation that only a portion of the destination state may ever be accessed, after a specific discrete transition. Thus, by *splitting the destination state* as well, based on what fraction of it is accessible from the

source state, we can refine the partitions. This simple extension was not necessary in linear systems, as the destination state space was typically entirely reachable from the resetted region (after a discrete transition). Since semi-algebraic sets have innate complexity, it is very unlikely that continuous evolution from different resetted regions will all envelope the entire state invariant. Clearly, since we chop off the region of the state invariant that is not reachable, the state invariants represent smaller sets. Hence, the extended-bisimulation-partitioning is likely to be a sharper one than the standard approach. The second advantage to this extended algorithm is that well-behaved subclasses can be characterized (see *Convergent Deterministic Automata* below). We now enumerate the complete series of computations:

Algorithm 1 [Extended Partitioning For Semi-Algebraic Automata]

(i) *Pick a state s (source) with a discrete transition to state d (destination);*

(ii) *Split s into two states s_d and $s_{\bar{d}}$ thus:* $Inv_{s_d}(X) \equiv \exists h, X' \ \langle s, X\rangle \xrightarrow[C]{h} \langle s, X'\rangle \wedge Guard_{s,d}(X')$ *and* $Inv_{s_{\bar{d}}}(X) \equiv Inv_s(X) \wedge \neg Inv_{s_d}(X)$;

(iii) *Split d is into d_s and $d_{\bar{s}}$ thus:* $Inv_{d_s}(X) \equiv \exists X'', X' \ Inv_s(X'') \wedge \langle s, X''\rangle \xrightarrow[D]{0} \langle d, X'\rangle \wedge \{\exists h \ \langle d, X'\rangle \xrightarrow[C]{h} \langle d, X\rangle\}$ *and* $Inv_{d_{\bar{s}}}(X) \equiv Inv_d(X) \wedge \neg Inv_{d_s}(X)$;

(iv) *The states $s_d, s_{\bar{d}}, d_s, d_{\bar{s}}$ replace s and d. The transition from s to d is replaced by the one from s_d to d_s. All other transitions to (or from) s (d) are each replaced by two transitions to (or from) s_d and $s_{\bar{d}}$ (d_s and $d_{\bar{s}}$);*

(v) *Repeat steps $(i) - (iv)$ until no transition from any state s to any state d can be found which splits s or d.* □

It is to be recalled that convergence of the partitioning does not imply decidability of reachability for general hybrid automata. As in the standard bisimulation case, the *over-approximated* set of points reachable from $\langle s_0, X_0\rangle$ in the original hybrid system is given by the *union of the invariants* of all the states along all the trajectories starting at the state d_0 of the partitioned system corresponding to the partition of s_0 containing X_0. Even in the non-convergent case, this procedure yields an estimate of the reachable set if we roll-out H for a reasonable number of steps. Similarly, we can check if a specific X_f is reachable from a specific X_0. We iteratively partition until the partition containing X_f is not in any trajectory starting from the partition containing X_0. We can then conclude guaranteed unreachability, or approximate reachability otherwise (counterexample-guided abstraction refinement).

Having generalized and extended an existing technique, we now characterize the broadest subclasses of hybrid systems where this new technique is well-behaved.

Convergent Deterministic Automata

In *deterministic* hybrid automata, a discrete transition is taken the moment its guard is satisfied (with no two guards ever holding simultaneously). Hence there is a unique future trajectory for every initial system state. If the *extended partitioning* procedure converges for a deterministic hybrid automaton, the original automaton will now correspond to a set of disconnected trajectories. Each of these will be a cycle of discrete states, with each state possibly preceded by a linear path of unique discrete states (all other topologies get excluded because there is no "future-branching" in deterministic automata). The extended partitioning *unlike the standard bisimulation partitioning*, produces exactly onto maps between successive states when convergent. We now show how many of their mathematical properties can be fruitfully exploited to address the reachability problem, for broad subclasses of convergent deterministic semi-algebraic hybrid automata.

In *linear convergent deterministic semi-algebraic automata*, all flows and reset maps are linear. Thus, infinite cycles are ruled out, since there are only a finite number of exactly onto maps possible (except when the sets have infinite axes of symmetry as does a circle). Thus:

Theorem 2.2 *There are only a finite number of 1-to-1 linear maps* $f(X) = \Sigma AX + B,\ A_i, B_i \in \mathbb{R}^d$ *possible, between two d-dimensional sets with finite axes of symmetry.* □

Corollary 2.3 *Given a cycle* $S_1, \cdots, S_n, S_1$ *of n d-dimensional sets with* s_l *axes of linear symmetry and* s_r *axes of rotational symmetry each, where each set maps exactly onto its successor* ($S_{i+1} = f(S_i)$)*, the number of unique successors of any point is at most* $ns_r 2^{s_l}$. □

Theorem 2.4 *Reachability over a deterministic semi-algebraic hybrid system with linear resets* $Reset_{u,v}(X, X') \equiv (X' = \Sigma AX + B),\ A_i, B_i \in \mathbb{R}^d$ *and linear flows* $Flow_u(X, X') \equiv (X' = \Sigma AX + B),\ A_i, B_i \in \mathbb{R}^d$ *is decidable, if the extended partitioning algorithm converges into states with finite axes of symmetry.* □

The more general notion of monotonicity has recently been identified as a useful restriction in characterizing hybrid systems [6]. A function is said to be monotonic (with respect to its arguments), if it is always increasing or always decreasing or constant in the specified interval. For *monotonic convergent deterministic semi-algebraic automata*, monotonic flow and reset maps guarantee that the system has to eventually converge to a fixed point or a limit cycle (chaotic behavior can be ruled out). We now show that, unlike linearity which ensures decidability of reachability, monotonicity only guarantees that approximate reachability can be decided upto any specified

accuracy. Thus:

Theorem 2.5 *If there exist 1-to-1 monotonic maps between two sets, all points converge to one of a finite number of fixed points or limit cycles.* □

Theorem 2.6 *Reachability over a deterministic semi-algebraic hybrid system with resets and flows monotonic (with respect to all system variables), that converges upon extended partitioning, is decidable up to an arbitrary degree of accuracy.* □

2.2 Approximating as a Polytope

The bisimulation approach produced a new hybrid system more amenable to approximate temporal analysis. A completely different approach, very popular for the reachability problem, is to approximate from the first step. This involves assuming a mathematically convenient geometrical shape for the initial set—the simplest being a polytope (bounded polyhedron), which can be written as a boolean combination of linear inequalities [3]. At each iteration, we compute the successor polyhedron by expanding it using the (one-step) transition relation of the hybrid system. Also, we need to ensure that the successor is also a polyhedron. At each iteration, the mathematics involves keeping track of the movement of the vertices and computing their new convex hull, or keeping track of the faces and moving them based on their maximum outward growth along the normal.

Clearly, a polyhedron can serve as a complexity restricting approximation of a semi-algebraic set as well. However, the conventional computational techniques are not applicable for two reasons. First, the *convex hull* of the successors of vertices of a polyhedron cannot be guaranteed to over-approximate the successor of the polyhedron. This is because, unlike linear systems, the flows cannot be assumed to be convexity preserving in semi-algebraic systems. Secondly, the *face-lifting* approach is not applicable in its basic from. This difficulty arises because, there is no straightforward way of calculating the maximum outward component of the flow along the normal to each face, of a polyhedron evolving with arbitrary polynomial dynamics.

In this section, we develop two new approaches that circumvent this problem. Instead of directly computing the approximated successor, we calculate the accurate complex successor (of the polyderon), and *then* approximate it with a new polyhedron. Though the accurate successor computation slows us down, it is still better than the entirely exact computation. This is because the quantified semi-algebraic expression for the successor is relatively simple (polyhedron). We first describe a very coarse over-approximation which merely keeps track of the extremities along each dimension. This simple over-

approximation can be obtained by calculating the maximum and minimum value along each dimension and bounding by one hyper rectangle. (We denote the value of the i-th dimension of X by X_i.)

Algorithm 2 [Over-Approximating as One Hyper-Rectangle]

(i) *Initialize the current over-approximation of the reachable set* $\mathcal{R}$ *with the starting hyper-rectangle* $\bigwedge_i(i_{min} \leq X_i \leq i_{max})$*;*

(ii) *Calculate the exact successor of* $\mathcal{R}$ *thus:*
$\mathcal{R}_E(\langle s, X\rangle) \equiv \exists s', X', h\ R(\langle s, X'\rangle) \wedge \langle s', X'\rangle \xrightarrow[T]{h} \langle s, X\rangle$*;*

(iii) *Calculate the maximum value of each dimension* X_i *in* $\mathcal{R}_E$ *thus:*
$\{\exists s, X\ (X_i = i'_{max}) \wedge \mathcal{R}_E(\langle s, X\rangle)\} \bigwedge \{\forall s, X\ \mathcal{R}_E(\langle s, X\rangle) \Rightarrow X_i \leq i'_{max}\}$*;*

(iv) *Calculate the minimum value of each dimension* X_i *in* $\mathcal{R}_E$ *thus:*
$\{\exists s, X\ (X_i = i'_{min}) \wedge \mathcal{R}_E(\langle s, X\rangle)\} \bigwedge \{\forall s, X\ \mathcal{R}_E(\langle s, X\rangle) \Rightarrow X_i \geq i'_{min}\}$*;*

(v) *For each dimension,* $i'_{min} \equiv min(i_{min}, i'_{min}),\ i'_{max} \equiv max(i_{max}, i'_{max})$*;*

(vi) *If* $j'_{max} \neq j_{max}$ *or* $j'_{min} \neq j_{min}$ *for some dimension* X_j*, repeat the steps* $(ii) - (v)$ *with* $\mathcal{R} \equiv \bigwedge_i(i'_{min} \leq X_i \leq i'_{max})$*; else, the procedure has converged.* □

While the utility of such a gross over-approximation is questionable, it is nevertheless a technique one can resort to when the complexity of the problem is very high. If we want to approximate with a general polyhedron (more than just a hyper-rectangle), we have to resort to the convex-hull or face-lifting approaches. As arbitrary face-lifting is not known to be amenable to computational analysis, we suggest a convex-hull-based approach. Since the new positions of the vertices cannot capture the new convex-hull, we move them by the maximum possible increments and decrements in one step of the hybrid system. In other words, we compute the maximum (and minimum) displacement (along each dimension) of any point in the polyhedron; and then assume that all the vertices could have moved by these amounts. The convex hull of the vertices, moved by these maximal amounts, is clearly guaranteed to be an over-approximation of the original polyhedron.

Algorithm 3 [Over-Approximating as One Hyper-Polygon]

(i) *Initialize the current over-approximation of the reachable set* $\mathcal{R}$ *with the starting hyper-polygon, composed of the initial set of* n *vertices* $v_1, \cdots, v_n$*;*

(ii) *Calculate the exact successor of* $\mathcal{R}$ *thus:*
$\mathcal{R}_E(\langle s, X\rangle) \equiv \exists s', X', h\ \mathcal{R}(\langle s', X'\rangle) \wedge \langle s', x'\rangle \xrightarrow[T]{h} \langle s, X\rangle$*;*

(iii) *Calculate the maximum increment* δ_{inc} *in each dimension* X_i *thus:*
$\{\exists s, X, s', X'\ \mathcal{R}(\langle s, X\rangle) \wedge \mathcal{R}_E(\langle s', X'\rangle) \wedge (X'_i - X_i = \delta_{inc})\} \bigwedge$

$\{\forall s, X, s', X'\ (\mathcal{R}(\langle s, X\rangle) \wedge \mathcal{R}_E(\langle s', X'\rangle)) \Rightarrow (X'_i - X_i \leq \delta_{inc})\}$;

(iv) *Calculate the maximum decrement* δ_{dec} *in each dimension* X_i *thus:*
$\{\exists s, X, s', X'\ \mathcal{R}(\langle s, X\rangle) \wedge \mathcal{R}_E(\langle s', X'\rangle) \wedge (X_i - X'_i = \delta_{dec})\} \bigwedge$
$\{\forall s, X, s', X'\ (\mathcal{R}(\langle s, X\rangle) \wedge \mathcal{R}_E(\langle s', X'\rangle)) \Rightarrow (X_i - X'_i \leq \delta_{dec})\}$;

(v) *Each vertex contributes* 2^d *new points, with each dimension being increased or decreased by the maximum amounts.* R *is assigned the convex hull of these* $n2^d$ *points;*

(vi) *Iterate* $(ii) - (vii)$ *until* $\delta_{inc} = \delta_{dec} = 0$. □

2.3 Rectangular Grid Abstraction

Instead of using one large polytope, the *grid abstraction* approach relies on keeping track of a number of small simple hyper-rectangles. Rectangular grids admit canonical representations and the number of faces grows linearly with the dimension, as opposed to convex polyhedra which become intractable in higher dimensions [2]. Two common simplifying strategies are restricting the vertices to be integers ("griddy") and the edges to be axis-parallel ("isothetic") [1].

We first show that the extension to semi-algebraic hybrid automata of the standard procedure is possible, because quantifier elimination can be used to compute the transitions between hyper-rectangles. One can partition the entire space into N^d hyper-rectangles, where N is the number of the $\mathcal{A}$-sized partitions [4] of each of the d dimensions. We use $B(X)$ to denote the k-dimensional grid unit $\bigwedge_i (B_i \leq X_i < (B_i + \mathcal{A}))$ of size $\mathcal{A}^d$. States will be connected to some of their $3^d - 1$ immediate neighbors, which differ by $+\mathcal{A}, -\mathcal{A}$ or 0 units in each dimension (with the identity-case alone excluded), and to some farther ones resulting from discrete resets. We now list the series of computations necessary to calculate the reachable region starting from a specific grid unit:

Algorithm 4 [Reachability Over Numerical Grids]

(i) *Given one hyper-rectangle* $F(X)$ *corresponding to the source;*

(ii) *Initialize "frontier" set* $\mathcal{F}$ *with* $\{F(X)\}$*, and "reachable set"* $\mathcal{R}$ *with* null*;*

(iii) *For each new hyper-rectangle* $P(X) \in \mathcal{F}$
  (a) *Compile the set of neighbors:*
  $\mathcal{N} \equiv \{Q(X) | (|Q_i - P_i| = \mathcal{A} \vee 0) \wedge \bigvee_i (|Q_i - P_i| \neq 0)\}$;
  (b) *For each neighbor* $Q(X)$ *of* $P(X)$ *not already in the reachable set,*

[4] $\mathcal{A}$ should be fixed in relation to the error in the $\xrightarrow[C]{h}$.

test if it is reachable i.e. $\exists X,\ P(X) \wedge \bigvee_{\forall v}\{\exists Y\ \bigvee_{\forall u}\langle v, X\rangle \xrightarrow[D]{0} \langle u, Y\rangle \wedge Q(Y)\}\ \bigvee\{\exists Y, h\ (0 < h \leq \mathcal{A}) \wedge \langle v, X\rangle \xrightarrow[C]{h} \langle v, Y\rangle \wedge Q(Y)$*;*

(c) *All candidate non-adjoint cells* $Q(X)$ *that can be reached by discrete state transitions can be tested thus:*
$\exists X,\ P(X) \wedge \bigvee_{\forall v}\{\exists Y\ \bigvee_{\forall u}\langle v, X\rangle \xrightarrow[D]{0} \langle u, Y\rangle \wedge Q(Y)\}$.

(d) *Add all reachable cells to both the reachable set* $\mathcal{R}$ *and the frontier set* $\mathcal{F}$ *and remove* $P(X)$ *from* $\mathcal{F}$*;*

(iv) *Iterate until there are no more new-hyper-rectangles.* □

Having shown that the standard procedure is applicable, we now develop a new approach for computing a sharper over-approximation (successor set of small hyper-rectangles) of the given hyper-rectangle. The idea is to compute the exact successor of a hyper-rectangle, and then over-approximate the region outside the initial hyper-rectangle (the "spill") by hyper-rectangles. In the previous case, we considered each of the $3^d - 1$ non-overlapping neighboring zones, and tested the transition to each. To simplify the expressions further, we suggest considering fewer overlapping neighbors; in particular, the zones with exactly one of the d dimensions increased or decreased i.e., $2d$ in all. To summarize, the standard method (previous case) accumulates the hyper-rectangles reachable from the given hyper-rectangle by testing transition to each of the $3^d - 1$ non-overlapping neighbors. The size of each neighbor is fixed ("griddy") forcing the approximation error to be at least that big. In the new technique, the hyper-rectangles continue to be axis-parallel ("isothetic"), but their vertices are not fixed. As a result, the approximation is guaranteed to be much better than the "griddy" case. The additional trick of considering fewer overlapping rectangles cannot be applied to the standard method, as the approximation will become too coarse.

We now present the details of the method. We estimate the spill in each neighboring zone by calculating the extremities in that zone, along the lines of the scheme for over-approximating the entire set as a single-hyper-rectangle. We use $B(X)$ to denote the k-dimensional grid unit $\bigwedge_i (B_i^l \leq X_i < B_i^r)$ (side of the hyper-rectangles are no longer fixed at $\mathcal{A}$). Further, $B_{\neg j,k}(X)$ denotes $\bigwedge_{i \neq j \vee k}(B_i^l \leq X_i < B_i^r)$.

Algorithm 5 [Approximating with Many Hyper-Rectangles]

(i) *As before, maintain the set of reachable hyper-rectangles* $\mathcal{R}$ *and the set of new hyper-rectangles* $\mathcal{F}$ *just added to the reachable set, representing the expanding frontier;*

(ii) *For each* $P(X) \in \mathcal{F}$*, compute the exact successor set of* $\mathcal{R}$ *thus:*

$$\mathcal{R}_E \equiv \bigvee_{\forall v}\{\exists Y \bigvee_{\forall u}\langle v, X\rangle \xrightarrow[D]{0} \langle u, Y\rangle \wedge P(Y) \bigvee\{\exists Y, h\ (0 < h \leq \mathcal{A}) \wedge \langle v, X\rangle \xrightarrow[C]{h} \langle v, Y\rangle \wedge P(Y);$$

(iii) *For each dimension X_i:*

(a) *For the neighbor $Q(X)$ where $Q_i^l = N_i^r$, calculate Q_i^r: $\{\exists X\ (P_{\neg i}(X) \wedge X_i = Q_i^r) \wedge \mathcal{R}_E(X)\} \bigwedge\{\forall X\ (P_{\neg i}(X) \wedge X_i > Q_i^r) \Rightarrow \neg\mathcal{R}_E(X)\}$. If $Q_i^r < P_i^r$, skip the next two steps;*

(b) *We now need to calculate the extremities l_j^{i+}, r_j^{i+} in each of the other dimensions X_j where $j \neq i$: $\{\exists X\ (P_{\neg i,j}(X) \wedge X_i > P_i^r \wedge X_i < Q_i^r \wedge X_j = l_j^{i+}) \wedge \mathcal{R}_E(X)\} \bigwedge\{\forall X\ (P_{\neg i,j}(X) \wedge X_i > P_i^r \wedge X_i < Q_i^r \wedge X_j < l_j^{i+}) \Rightarrow \neg\mathcal{R}_E(x)\}$ and $\{\exists X\ (P_{\neg i,j}(X) \wedge X_i > P_i^r \wedge X_i < Q_i^r \wedge X_j = r_j^{i+}) \wedge \mathcal{R}_E(X)\} \bigwedge \forall X\ (P_{\neg i,j}(X) \wedge X_i > P_i^r \wedge X_i < Q_i^r \wedge X_j > r_j^{i+}) \Rightarrow \neg\mathcal{R}_E(X)$.*

(c) *The hyper-rectangle defined by $Q_i^l < X_i < Q_i^r \bigwedge_{j\neq i} l_j^{i+} < X_j < r_j^{i+}$ is added to the list of new hypercubes and also to the reachable set R;*

(d) *Repeat the above three steps for the neighbor where $Q_i^r = P_i^l$ and $Q_i^l, l_j^{i-}(< X_j), (X_j <) r_j^{i-}$ need to be calculated;*

(iv) *Repeat $(ii) - (iii)$ until the procedure converges.* □

In the "griddy" case, we inspect every possible neighbor and test transition. Alternately, we could have computed the exact successor of the entire set, and then extracted the component hyper-rectangles. Such an approach would require a procedure for converting a semi-algebraic set (the exact successor) into an over- (or under-) approximating union of hyper-rectangles of fixed dimension.

In the "isothetic" case, we over-approximated the "spill" outside the hyper-rectangle with a hyper-rectangle in each neighboring zone (with substantial overlap). Alternatively, we could compute the best non-overlapping but non-griddy hyper-rectangles that cover the newly reachable points, without having to compute the maximum and minimum values of each dimension in each neighboring zone. This approach again requires a general procedure for converting the exact successor into a union of hyper-rectangles of arbitrary dimension.

We solve this problem by actually testing if potential vertices (from a griddy or isothetic grid) are included in the exact reachable set. We then use the resulting set of present and absent points to pick candidate hyper-rectangles. Quantifier elimination is still necessary, since we may wish to guarantee that the hyper-rectangles we have picked are wholely inside (under-approximation) or that the hyper-rectangles we have omitted are wholely outside (over-approximation). Hence the approach we have suggested addresses

the problem of minimizing the number of quantifier-elimination queries. We now provide the details of this new technique, which can be used in conjunction with both the algorithms presented before.

Algorithm 6 [Over-Approximating using Hyper-Rectangles]

(i) *Calculate* i_{max} *and* i_{min}*, the maximum and minimum values of* X_i *in the given set* $\mathcal{R}$*:* $\{\exists X\ (X_i = i_{max}) \wedge \mathcal{R}(X)\} \bigwedge \{\forall X\ \mathcal{R}(X) \Rightarrow X_i \leq i_{max}\}$ *and* $\{\exists X\ (X_i = i_{min}) \wedge R(X)\} \bigwedge \{\forall X\ R(X) \Rightarrow X_i \geq i_{min}\}$*;*

(ii) *Split each dimension into equidistant points of the desired resolution;*

(iii) *Evaluate membership in* R *for each grid point* g *by substitution:* $\mathcal{R}(g)$*;*

(iv) *The small hyper-rectangles created by the grid points which contain at least one vertex in* $\mathcal{R}$ *are immediately included in the over-approximation;*

(v) *Hyper-rectangles where none of the vertices are in* $\mathcal{R}$ *are included only if* $\exists x \in G\ \mathcal{R}(x)$ *returns* true. □

In the under-approximation case, hyper-rectangles with at least one vertex not in R can be safely omitted. The hyper-rectangles with all vertices in R are the contenders for quantifier elimination. In both cases, one could use a "proof-by-example" approach, where one verifies the feasibility at some randomly selected points (center being the first choice) to see if quantifier elimination can be avoided. By randomizing or biasing the grid points, one can obtain non-griddy vertices. If, in addition, high-dimensional convex hull algorithms are used, one could build upon this method to derive general polyhedral representations as well.

2.4 Time Discretization

For the sake of completeness, we also note that time discretization can be employed (in conjunction with most techniques) to approximate the hybrid system dynamics. Conventionally, the most restricted transition relation enforces continuous evolution for a fixed time-step Δ followed by one optional discrete transition. The typical "improvement" over the previous case could be allowing the discrete jump anywhere during the continuous evolution, as opposed to only at the end of it. This model could be made even more realistic by allowing N jumps anywhere during the continuous evolution. Clearly, the only paths that get excluded here are those that involve more than N jumps in Δ time. All the restrictions described above are "fixed step" i.e. the system progresses in timesteps of Δ. Each of them could be relaxed by allowing the time-step to be in the range $[0, \Delta]$ to capture many other behaviors. Such restrictive transition relations greatly simplify fixpoint evaluations of temporal logic operators.

A completely different time-discretization-based under-approximating approach would be to ignore the behavior of the system *during* the continuous evolution. We simply use the end-points to verify the temporal query. For example, the TCTL one-step until operator for semi-algebraic hybrid systems [8] can be simplified as: $p \triangleright q = q \bigvee_{\forall v}\{\exists s \bigvee_{\forall u}\langle v,r\rangle \rightarrow_D^0 \langle u,s\rangle \wedge q(s)\} \bigvee\{\exists s,h\ (0<h\leq\Delta)\wedge\langle v,r\rangle \rightarrow_C^h \langle v,s\rangle \wedge q(s) \wedge p(r)\}$. Another simplifying over-approximation would be to assume that the state invariant needs to be true only at the beginning of (and not all along) the Δ time units of continuous evolution. This heuristic could prove particularly useful if we combine time discretization with the partitioning algorithm discussed earlier (which will accumulate complex state-invariants).

3 Discussion

In this paper, we have extended the theory of approximate verification of hybrid systems from the linear to the more expansive semi-algebraic domain. The algebraic model checking method presented in [8] was made more computationally practicable by extended bisimulation partitioning, approximation with general polyhedra and unions of simple polyhedra, and time discretization. For the extended bisimulation procedure suggested, we identified well-behaved subclasses based on some novel critical observations about the behavior of exactly onto linear and monotonic maps between arbitrary sets. For polyhedral approximations, we used the maximum and minimum values of the system variables and their possible growth in one step to expand the convex hull. We demonstrated how these same metrices (maximal growth along each dimension in one step) could be used to obtain a hyper-rectangular approximation of semi-algebraic sets. We also introduced a practical strategy to identify candidate hyper-rectangles, for which the quantifier elimination needs to be invoked. Time discretization was seen to simplify the problem by allowing fewer discrete jumps, excluding zeno paths, and by verifying the temporal property at certain sampling times rather than everywhere.

All these methods need to be refined to better handle discrete resets and symbolic approximations. More crucial is their actual implementation and performance analysis. On the purely algebraic side, approximate quantifier elimination and direct maximum-minimum estimation of a semi-algebraic set are those mathematical techniques, that need to be developed to further accelerate these methods. There are several approximating methods that are yet to be extended to semi-algebraic systems. These include: (1) piece-wise approximations of continuous dynamics; (2) problem domain transformation: optimal control using Pontryagin Maximum Principle, level sets of solutions to

Hamilton-Jacobi-Bellman equations, sum of squares decomposition (semidefinite programming) and geometric programming; (3) predicate abstraction and qualitative simulation; (4) other geometric approximations: oriented rectangular hulls, zonotopes and ellipsoids. Next, we wish to determine practical applicability of these methods, trade-offs among them and suitable combinations of these approximations that work best with the available tools.

References

[1] E. Asarin, T. Dang, O. Maler, and O. Bournez. Approximate Reachability Analysis of Piecewise-Linear Dynamical Systems. In B. Krogh and N. Lynch, editors, *Hybrid Systems: Computation and Control (HSCC'00)*, volume 1790 of *LNCS*, pages 20–31. Springer-Verlag, 2000.

[2] O. Bournez, O. Maler, and A. Pnueli. Orthogonal Polyhedra: Representation and Computation. In F. Vaadrager and J. van Schuppen, editors, *Hybrid Systems: Computation and Control (HSCC 1999)*, volume 1596 of *LNCS*, pages 19–30. Springer-Verlag, 1999.

[3] A. Chutinan and B. Krogh. Verification of Polyhedral-Invariant Hybrid Automata Using Polygonal Flow Pipe Approximations. In F. W. Vaandrager and J. H. van Schuppen, editors, *Hybrid Systems: Computation and Control (HSCC'99)*, volume 1569 of *LNCS*, pages 76–90. Springer-Verlag, 1999.

[4] Ronojoy Ghosh and Claire Tomlin. Symbolic reachable set computation of piecewise affine hybrid automata and its application to biological modelling: Delta-notch protein signalling. *Systems Biology*, 1(1):170–183, June 2004.

[5] H. Hong. Quantifier elimination in elementary algebra and geometry by partial cylindrical algebraic decomposition, version 13. *WWW site* www.eecis.udel.edu/~saclib, 1995.

[6] R. Lanotte and A. Maggiolo-Schettini. Monotonic hybrid systems. *Journal of Computer and System Sciences*, 2004.

[7] B. Mishra. *Computational Real Algebraic Geometry.* CRC Press, Boca Raton, FL, 2004.

[8] V. Mysore, C. Piazza, and B. Mishra. Algorithmic algebraic model checking II: Symbolic bounded model checking and reachability results. In *Automated Technology for Verification and Analysis (ATVA) (submitted)*, 2005.

[9] C. Piazza, M. Antoniotti, V. Mysore, A. Policriti, F. Winkler, and B. Mishra. Algorithmic algebraic model checking I: The case of biochemical systems and their reachability analysis. In *Computer Aided Verification (CAV)*, 2005.

[10] A. Tarski. *A Decision Method for Elementary Algebra and Geometry.* University of California Press, second edition, 1948.

4 Appendix

4.1 Semi-Algebraic Hybrid Automata: Definitions and Decidability

Definition 4.1 Semi-Algebraic Set[7] Every quantifier-free boolean formula composed of polynomial equations and inequalities defines a semialgebraic set (i.e., unquantified first-order formulæ over the reals —$(\mathbb{R}, +, \times, =, <$)). □

Definition 4.2 Semi-Algebraic Hybrid Automata [9] A k-dimensional *hybrid automaton* is a 7-tuple, $H = (Z, V, E, Init, Inv, Flow, Jump)$, consisting of the following components:

- $Z = \{Z_1, \ldots, Z_k\}$ and $Z' = \{Z'_1, \ldots, Z'_k\}$ are two finite sets of variables ranging over the reals $\mathbb{R}$
- (V, E) is a directed graph of discrete states and transitions
- Each vertex $v \in V$ is labeled by "Init"(initial), "Inv"(invariant) and "Flow" labels of the form $Init_v[Z]$, $Inv_v[Z]$, and $Flow_v[Z, Z', t, h]$
- Each edge $e \in E$ is labeled by a "Jump" condition of the form $Jump_e[Z, Z'] \equiv Guard_e(Z) \wedge Reset_e(Z, Z')$
- *Init*, *Inv*, *Flow*, and *Jump* are semi-algebraic. □

Definition 4.3 Semantics of Hybrid Automata[8] Let $H = (Z, V, E, Init, Inv, Flow, Jump)$ be a hybrid automaton of dimension k.

- A *location* ℓ of H is a pair $\langle v, R\rangle$, where $v \in V$ is a state and $R \in \mathbb{R}^k$ is an assignment of values to the variables of Z. A location $\langle v, R\rangle$ is said to be *admissible*, if $Inv_v(R)$ is satisfied.
- The *continuous reachability transition relation* $\xrightarrow[\mathcal{C}]{h}$ forces the state invariant to hold at every point except the end-point along the evolution curve determined by the flow equations during the $h(> 0)$ time units from the current time t_0:

$$\langle v, R\rangle \xrightarrow[\mathcal{C}]{h} \langle v, S\rangle \quad \text{iff}$$
$$\Big(Flow_v(R, S, t_0, h) \ \wedge \ \forall Z', h' \in [0, h)\ Flow_v(R, Z', t_0, h') \Rightarrow Inv_v(Z') \Big),$$

  where $Flow_v(Z, Z', T, h)$ is the flow label of v.
- The *discrete reachability transition relation* $\xrightarrow[\mathcal{D}]{0}$ ensures that both parts of the *zero-time* jump – the guard condition which needs to be satisfied just before the transition is taken, and the reset condition which determines the values after the transition, are satisfied.

$$\langle v, R\rangle \xrightarrow[\mathcal{D}]{0} \langle u, S\rangle \quad \text{iff} \quad \langle v, u\rangle \in E \ \wedge \ Jump_{v,u}(R, S).$$

- The *transition relation* $\mathcal{T}$ of H connects the possible values of the system variables before and after *one step*—a discrete step for a time $h = 0$ or a continuous evolution for any time period $h > 0$:

$$\mathcal{T}(\ell \xrightarrow{h} \ell') = \ \{h = 0 \wedge \ell \xrightarrow[\mathcal{D}]{0} \ell'\} \ \vee \ \{h > 0 \wedge \ell \xrightarrow[\mathcal{C}]{h} \ell'\}.$$

- A *trace* of H is a sequence $\ell_0,\ell_1, \ldots, \ell_n, \ldots$ of admissible locations such that

$$\forall i \geq 0,\ \exists h_i \geq 0,\ \mathcal{T}(\ell_i \xrightarrow{h_i} \ell_{i+1}). \quad \square$$

Remark 4.4 When a semi-algebraic relation $Flow_v(R, S, t, h)$ is used between the continuous states R at time t and S at time $t + h$ in a discrete state v, it may have been "derived" in two ways: (1) *Solution Is A Polynomial*: The equation describing the continuous evolution of the variables in a discrete state is a polynomial, say $Y(t)$, and $Flow_v(Z, Z', t, h) \equiv \{\ Z = Y(t)\ \wedge\ Z' = Y(t + h)\ \}$. Or, (2) *Differential Equation Is A Polynomial*: Differential equations describing the continuous evolution are *approximated* in $Flow_v$ using one of the symbolic integration schemes (e.g., the *Taylor* series in [9] or based on a direct integration scheme such as the linear *Euler* or the higher degree *Runge-Kutta*). The error is controlled by an upper bound (say Δ) on the time spent in one continuous step as we aim for over- or under-approximating the flow equations. The Lagrange Remainder Theorem can be used to estimate errors.

Definition 4.5 $\triangleright$ for Semi-Algebraic Hybrid Systems. The expression $p \triangleright q$ is *True* at the current continuous state R if q is true now, *OR*

- For one of the possible current discrete states v, there exists at least one state u to which a transition can be taken such that q holds at the end, *OR*
- For one of the possible current discrete states v, there exists a continuous transition (of at most Δ time units when we need to upper-bound the flow-approximation error) all along which $p \vee q$ holds, with q being true at the end.

$$\begin{aligned} p \triangleright q \ =\ & q(R) \bigvee_{\forall v} \\ & \Big(\{\exists S\ \bigvee_{\forall u} \langle v, R\rangle \xrightarrow[\mathcal{D}]{0} \langle u, S\rangle \wedge q(S)\}\ \bigvee \\ & \{\exists S, h\ (0 < h \leq \Delta) \wedge \langle v, R\rangle \xrightarrow[\mathcal{C}]{h} \langle v, S\rangle \wedge q(S)\ \wedge \\ & \quad \forall S', h'\ ((0 \leq h' < h) \wedge \langle v, R\rangle \xrightarrow[\mathcal{C}]{h'} \langle u, S'\rangle) \Rightarrow (p(S') \vee q(S'))\} \Big) \quad \square \end{aligned}$$

Remark 4.6 The last term in the formula, $p(S') \vee q(S')$, can be replaced with just $p(S')$ for evaluating $\exists \mathcal{U}$ over semi-algebraic hybrid systems. Also, the upperbound Δ on h should be *omitted* if there is no error in the $Flow_v$ expression.

Theorem 4.7 *[8] The* one-step-until *operator $p \triangleright q$ is decidable for semi-algebraic hybrid systems if p and q are also semi-algebraic.*

Corollary 4.8 *For semi-algebraic hybrid systems:*

(i) $\exists\mathcal{U}$, $\exists\mathcal{F}$, $\exists\mathcal{G}$ *and their subscripted versions* $\exists\mathcal{U}_{\leq z}$, $\exists\mathcal{F}_{\leq z}$ *and* $\exists\mathcal{G}_{\leq z}$ *are* semi-decidable.

(ii) *The* negations *of* $\forall\mathcal{U}$, $\forall\mathcal{F}$, $\forall\mathcal{G}$ *and their subscripted versions* $\forall\mathcal{U}_{\leq z}$, $\forall\mathcal{F}_{\leq z}$ *and* $\forall\mathcal{G}_{\leq z}$ *are* semi-decidable.

(iii) *All subscripted operators become* decidable *in the absence of* zeno *paths.*

4.2 Details of Proofs

Proof Of Theorem 2.1 Each state s (source) needs to be split into two states s_d and $s_{\bar{d}}$ depending on whether or not the guard of the transition to each d (destination) can ever be satisfied. Since real quantifier elimination is decidable [10,5], these partitions can be computed thus: $Inv_{s_d}(X) \equiv \exists h, X' \langle s, X\rangle \xrightarrow[C]{h} \langle s, X'\rangle \wedge Guard_{s,d}(X')$ and $Inv_{s_{\bar{d}}}(X) \equiv Inv_s(X) \wedge \neg Inv_{s_d}(X)$.□

Proof Of Theorem 2.2 Consider two sets S_1 and S_2 between which *onto* linear maps exist. Maps of the form $x' = \Sigma a_i x_i + a_0$ correspond to a rotation, stretch and shift of the coordinate axes. In other words, S_2 *has* to be a stretched, rotated and shifted image of S_1 for such an onto map to exist. There are 2^{s_l} maps possible because of the s_l axes of linear symmetry, on *each* of the s_r axes of rotational symmetry. Hence the total number of coupled linear onto maps is $s_r 2^{s_l}$. □

Proof of Corollary 2.3 Let $m(= s_r 2^{s_l})$ be the number of possible onto maps between S_1 and S_2. Let $x_1 \in S_1$ map to one of $y_1, \cdots, y_m \in S_2$. Let y_1 map to one of $x_1, \cdots, x_m \in S_1$ (x_1 has to appear because the inverse of a linear map is linear). Suppose x_2 maps to y_{m+1}. Since linear maps are closed under composition and retain their ontoness property, by following the linear maps from $x_1 \rightarrow y_{1\leq i\leq m} \rightarrow x_{1\leq i\leq m} \rightarrow y_{m+1}$, we get a new linear map that takes x_1 to y_{m+1}. However, we know from symmetry arguments that only m linear maps can exist. This contradiction proves that no matter how many times we compose the two given onto linear maps, we remain within the set of $2n$ points (for example, rectangles have 8 possible linear onto maps while cubes have 24). Extending this argument to a cycle of n states, each point can have only one of m different successors in each of the n states. Hence, the length of the biggest cycle is nm. □

Proof of Theorem 2.4 The continuous evolution can be treated as a linear map from the initial value to the final value that first satisfies the guard. Further, *time* does not appear in the equation as in deterministic systems, an initial value corresponds to a unique final value. No restriction on the guard is necessary as we assume there is only 1 successor to each discrete state. Thus

a cycle of n states corresponds to a cycle of $2n$ linear 1-to-1 maps with only the values before and after a reset sufficing to capture the dynamics. If m is the maximum number of possible onto maps between any two consecutive sets, the number of unique successors is $\leq 2nm$. Since x_0 has a finite number of successors, if the target x_f is not reached before the system begins to cycle, we conclude exact unreachability. If x_f is eventually reached, then it is indeed exactly reachable. □

Proof of Theorem 2.5 Let the sequence be $X_0,\ X_1 = f(X_0),\ X_0' = g(X_1) = g(f(X_0)), \cdots$. Since f and g are monotonic, X will continue to move in the same direction. The process can continue ad infinitum if X approaches a fixed point ($g(f(X)) = X$). The other mathematical alternative is that it approaches a limit cycle ($f(g(f(g(..(X)..)))) = X$). Monotonicty ensures progress while ontoness ensures finiteness of the number of iterations required to reach the neighborhood of a fixed point or a limit cycle. □

Proof of Theorem 2.6 Just as in the linear case, the continuous flow can also be thought of as a monotonic function from the initial value (from a reset that brought the system to this state) to the final value (when a guard is first satisfied). Thus any cycle of n discrete states corresponds to a cycle of $2n$ monotonic maps (n flow-maps and n reset maps). Further, from the previous theorem, we know that iterative evolution along such a cycle of exactly onto maps has to approach a fixed point or a limit cycle. We can stop iterating when $\bigwedge(|X_i - X_i'| < \epsilon_i$, where X is the d-dimensional value of the system variables, ϵ_i is the desired accuracy in the i-th dimension and X' is the value after one cycle (reached the neighborhood of a fixed point) or after $2, 3, \cdots, d$ cycles (reached the neighborhood of a limit cycle) . The monotonic resets guarantee that this will happen in a finite number of steps and that once this happens, the system cannot escape out of it. □